



Article

Software-Bus-Toolchain (SBT): Introducing a Versatile Method for Quickly Implementing (I)IoT-Scenarios

Simon D. Duque Anton

comlet Verteilte Systeme GmbH, Amerikastr. 27, 66482 Zweibruecken, Germany; simon.duque-anton@comlet.de

Abstract: The Internet of Things (IoT) has become ubiquitous. IoT devices are applied in a multitude of applications, e.g., in smart home scenarios, building automation, smart energy and smart cities, healthcare, and industrial environments. Fast and efficient implementation and roll-out of IoT devices is a critical factor for success and acceptance of IoT devices. At the same time, the variety of hardware platforms that can be used for IoT applications, as well as the number of IoT orchestration platforms is increasing. Finding the right combination of tooling and hardware is not trivial, but essential for building applications that provide value. In this work, a Software-Bus-Toolchain (SBT) is introduced that encapsulates firmware design, data point definition, and communication protocol usage. Furthermore, an IoT control platform is provided to control and evaluate the IoT modules. Thus, using the SBT, solely the business logic has to be designed, while the hardware-design is automated to a high degree. Usage of the Zephyr framework allows the interchange of hardware modules, while interfaces provide easy adaption of data points and communication capabilities. The implementation of interfaces to the IoT-platform as well as to the communication layer provides a universal usage of logic and data elements. The SBT is evaluated in two application scenarios, where its flexible nature is shown.

Keywords: (Industrial) Internet of Things; Software-Bus; IoT-platform; communication stack

**Citation:** Anton, S.D.D.Software-Bus-Toolchain (SBT):
Introducing a Versatile Method for
Quickly Implementing (I)IoT-Scenarios.
Future Internet **2024**, *16*, 237. <https://doi.org/10.3390/fi16070237>Academic Editors: Giovanni Pau,
Marco Lombardi, Domenico
Santaniello and Mario Casillo

Received: 25 April 2024

Revised: 21 June 2024

Accepted: 24 June 2024

Published: 3 July 2024



Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The term Internet of Things (IoT) gained popularity after Kevin Ashton used it in 1999, in conjunction with the development of Radio Frequency IDentification (RFID) technology [1]. It is used for sensor and actuator networks, usually employed in a distributed manner with a high data volume, using Internet Protocol (IP)-based technologies, e.g., as defined by McKinsey [2]. These definitions imply that the value of IoT networks does not lie in their hardware-capabilities alone, but in the possible combinations of sensors and actuators to obtain multi-dimensional information in the application environment. Consequently, the market for IoT applications is consistently increasing, with a peak of a global market size of 384.70 billion USD in 2021 according to Fortune Business Insights™ [3]. A large number of different application areas explain the constant increase in demand. End users employ IoT devices in smart home environments [4,5]. The principal of control, sensing and automation with distributed devices is employed in smart city [6,7], smart farming [8,9] and industrial [10] applications. Smart grids [11] assist in the smart and sustainable distribution of energy. Retail [12,13] scenarios with IoT assistance provide a more convenient shopping experience, while IoT in healthcare [14] increases efficiency in treating and curing people. In general, IoT applications provide the potential to lower cost and management effort by minimising the need for on-site operations while increasing efficiency by providing insights from measurements and data aggregation. This potential provided by IoT solutions has risks, however. Security is paramount, as vulnerabilities in systems interacting with the physical world provide threats to digital as well as physical assets [15,16]. Generally, IoT devices are expected to be cheap, with a short time to market [17]. This puts constraints on the development and integration of such devices, leading

to code reuse and a consequent vulnerability of devices to the same threats [18]. A strong competition has emerged on the IoT market, highlighted by the growing number “publicly known IoT platforms”, as analysed by IoT Analytics [19]. As depicted in Figure 1, provided by IoT Analytics (<https://iot-analytics.com/iot-platform-companies-landscape/>, accessed on 23 June 2024) [19].

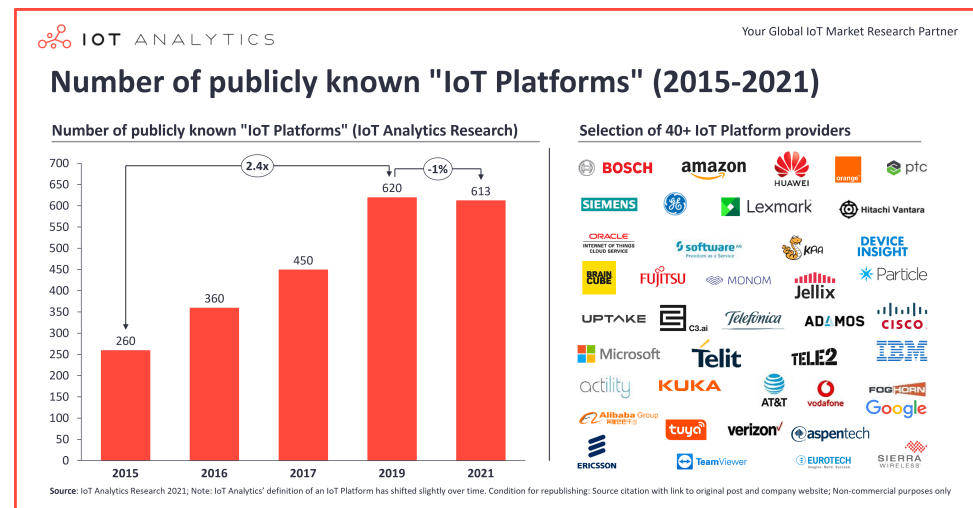


Figure 1. Development of IoT platforms according to IoT Analytics [19].

An increase in IoT platforms between 2015 and 2019 can be observed. However, starting 2019, the trend stops and the number of IoT platforms remains stagnant. This fact, combined with the ongoing growth of the IoT market size indicates that IoT is still a sound business model. According to their study, a number of 188 platform operators has ceased business over the years 2019 and 2020. Such platforms are commonly used to orchestrate and manage IoT devices. Apart from the platform, the devices are required. Low-cost embedded hardware platforms are commonly used, providing a number of development environments for easy implementation and set-up, such as Arduino [20], MIKROE [21] and Zephyr [22]. These tools still require a significant amount of coding, but mitigate the need to program hardware bare-metal. The large number of management platforms on the one hand and the still high implementation effort on the hardware-side motivate the need for an end-to-end IoT builder framework that allows an easy implementation as well as set-up and management of devices.

In addition to the IoT employed in home environments, the Industrial Internet of Things (IIoT) has been introduced to industrial applications as part of the concept of Industry 4.0. Despite the difference in application scenarios, the use cases are often similar and include distributed, heterogeneous systems that need to provide correct information in a timely matter. While Open Platform Communication Unified Architecture (OPC UA) has become the de facto standard for industrial communication in the context of Industry 4.0, an abundance of scenarios still employ legacy communication protocols.

In the course of research project InnoTop, a Software-Bus-Toolchain (SBT) was developed to meet this demand. InnoTop is a public funding programme provided by the Federal State Rhineland Palatinate and managed by the Investitions- und Strukturbank Rheinland-Pfalz (ISB). Its aim is to foster research and development in local SMEs. The contribution of this work consists of the development of an IoT builder framework that:

- Allows for a quick implementation of business logic. That means the actual functionality in should be easy to set up without implementation of low-level functions.
- Functionally separates data elements the IoT device interacts with, meaning the data can be defined and used independent from the low-level implementation.
- Functionally separates the communication protocol from the business logic implementation.

This framework allows for the quick creation of novel IoT devices and networks as well as integration of new IoT devices into existing environments, resulting in less time to release, less low-level implementation effort, higher flexibility in terms of decoupling hardware from function and unified data elements. In contrast to existing tools, the SBT considers programming environment for embedded devices and IoT platform alike, while reducing implementation effort for both. The implementation of abstraction layers allows for seamless transfer of application code between devices. Furthermore, these abstraction layers enable the integration into an IoT platform. Data is defined and standardised into data points that are described as Google Protocol Buffer elements. This standardisation provides fast creation and re-use of elements as well as a standardised interface to the communication layer. Additionally, any computation on the IoT platform, for which Node-RED is used, is performed based on these data points by functions specifically implemented in the course of the project. In the course of project SBT wrapper functions are implemented that provide standardised interfaces and enable exchange of data from physical sources, as well as hardware platform.

The remainder of this work is structured as follows. A background is provided in Section 2. Requirements for an IoT builder set up are discussed in Section 3. The components developed in the course of this work are presented in Section 4. Implementation details are introduced in Section 5. The IoT builder environment is evaluated in Section 6. A conclusion is drawn in Section 7.

2. Background

Two different aspects of frameworks for IoT environments are discussed in this section: Management and orchestration approaches discussed in the scientific community, and management and control platforms that are predominately available in commercial and open source software projects. Noura et al. evaluate taxonomies and present levels of interoperability that can be used to measure the usefulness of IoT systems [23]. They distinguish between the following six levels of interoperability:

- Device Level Interoperability
- Network Level Interoperability
- Syntactic Level Interoperability
- Semantic Level Interoperability
- Cross-platform Interoperability
- Cross-domain Interoperability

These levels of interoperability are used to evaluate the use cases that a certain IoT solution can satisfy the requirements for.

Braten et al. present a survey on autonomous IoT device management systems [24]. They distinguish along several dimensions and classify scientifically developed approaches to IoT device management according to several adaption mechanisms, where the functional aspects of the device management are considered. Cheruvu et al. present a survey of IoT frameworks with a focus on complexity in terms of functionality and resource cost [25]. Several authors address the demands posed by the heterogeneity of IoT networks as well as simplified representation of services and devices. Aïssaoui et al. introduce an ontology-based approach to handling heterogeneous IoT devices [26]. Privat discusses how IoT devices can be integrated into a broader network context and used for complex sensor and actuator networks [27]. Fortino et al. discuss the digital representation and integration of IoT objects into digital libraries for simple reuse and management [28]. Microservice-based architectures are discussed as well. Uviase and Kotonya survey existing integration frameworks for Service Oriented Architectures (SOAs) in IoT and conclude with a microservice-based architecture for enhanced interactionability [29]. Similarly, the interoperability of IoT devices in a Peer to Peer (P2P) fashion is discussed by Chen [30]. Sun et al. present a microservice-based architecture for IoT networks [31]. Furthermore, resource constraints and Wireless Sensor Networks (WSNs) are commonly discussed. Mavromatis et al. highlight the networking aspect of IoT device management [32]. They focus on edge

and cloud approaches with Software-Defined Network (SDN)-based solutions in WSNs. In their Machine to Machine (M2M) scenario, this communication provides a significant decrease in traffic volume. Maloney et al. discuss an agent-based approach to address the resource constraints in IoT devices [33]. They aim at mitigating the overhead in commonly used management platforms with a lightweight mechanism. Großmann et al. discuss a sensor monitoring framework that provides a resource-efficient platform for collecting information from sensor devices [34]. IoT can be applied in several domains. Calderoni et al. introduce an IoT management platform focussing on smart cities [35]. Balachandar and Chinnaiyan address the security aspects of IoT networks with a rule-based approach that allows management of the security objectives [36]. An application for IoT in healthcare is discussed by Gelogo et al. [37]. Burger et al. present an approach for integrating Field-Programmable Gate Arrays (FPGAs) into IoT networks in an on-demand-basis to better distribute and manage resources [38]. Perumal et al. present a concept architecture for smart home IoT applications [39]. It basically consists of wrappers that identify devices in a network and map their functionality to actionable items in the presented framework. Implementation of IoT devices is not discussed in their paper.

Management and orchestration platforms are provided by major cloud providers, such as Microsoft [40], IBM [41], GE [42], and Amazon [43]. Bilgeri et al. present an IoT business model builder approach in a white paper [44]. They put a strong focus on the organisational aspects in creating business value from IoT systems. Apart from the large companies, there are several smaller companies, such as twilio [45], openremote [46], or MiMove [47]. The main concern of such platforms is the management of devices, i.e., the collection of data and provision of commands. They offer respective eco-systems that allow integration of novel, pre-conditioned devices into a system. However, the development and implementation of devices is mostly not considered. This is a drawback which is considered by the SBT framework. Platforms such as macchina [48], DeviceHive [49], DSA [50], Kaa [51], and openHAB [52] provide platforms for easy integration and control of devices. These devices need to be able to communicate with the platforms in order to function correctly. On the other hand, there are frameworks, such as Zephyr [22] and Arduino [20] that focus on simple implementation of IoT devices. In a similar fashion, Zetta provides interfaces for IoT components to enable easy integration of devices into IoT systems [53]. The SBT framework presented in this work employs Zephyr for creation of IoT devices.

3. IoT-Builder Requirements

There is a consistency in literature regarding IoT products about the obstacles to overcome in order to obtain sound and useful IoT products [54,55]. Commonly mentioned criteria that need to be addressed for successful implementation of products contain:

- Security,
- connectivity,
- heterogeneity and cross-platform compatibility,
- data collection and processing, and
- the skill set of the people.

Additionally, Hackbarth mentions the three challenges of

- “Rapid application development for IoT”,
- “Managing heterogeneity and diversity”, and
- “Building customizable IoT solutions”

In a blog contribution for Bosch [56]. These assessments from different sources highlight that developing and rolling out IoT products requires diverse expertise and resources. The devices have to be implemented on a variety of hardware bases, be integrated into IoT platforms and exchange a well-defined set of data and controls. Thus, IoT implementation requires knowledge in hardware-constrained programming for embedded devices, data engineering and management as well as platform architectures. Due to the hardware used for IoT-scenarios, resource constraints are to be taken into account as well. In addition

to these conditions, timing commonly is a factor. Embedded hardware boards that provide the basis for IoT devices have short update cycles. Communication protocols are regularly introduced and need to be integrated into new and existing platforms. Based on these restrictions, the complete SBT toolchain adheres to certain requirements. The SBT toolchain needs to be extensible, i.e., if new hardware boards or communications protocols are introduced, they are easily integratable into the platform. Use cases need to be easily implemented with the SBT toolchain. That necessitates a high level of abstraction so that commonly used functionality is handled independent of the user. This additionally allows for a separation of business case and technical implementation which reduces time to market and implementation effort. Furthermore, as an enabling condition for this requirement, data and communication needs to be abstracted and provided via a defined interface for an abstracted and pre-defined control platform. How these requirements are met is discussed in the next section. After that, demonstrator implementations of IoT networks based on the SBT toolchain are presented.

4. Software-Bus Components

The SBT consists of several components that are combined to either individual devices or extensive networks. These components address every aspect in IoT architectures. They are discussed in the following subsections.

4.1. Architecture and Structure

Regarding the structure of the SBT, two aspects need to be considered: The implementation part as well as the operation part. During implementation, the SBT provides extensive aid in setting up parts of or complete IoT environments. During operation, the architecture can be employed both in green field and brown field scenarios, i.e., SBT devices can be integrated into existing networks or used to create networks from scratch. Regarding the implementation of IoT devices or environments, SBT offers a tool box that provides data point descriptions and communication stacks which can be used with a variety of different hardware devices. Setting up a device or network only requires implementation of business logic as functional aspects of data description and communication are addressed by SBT. This approach is depicted in Figure 2.

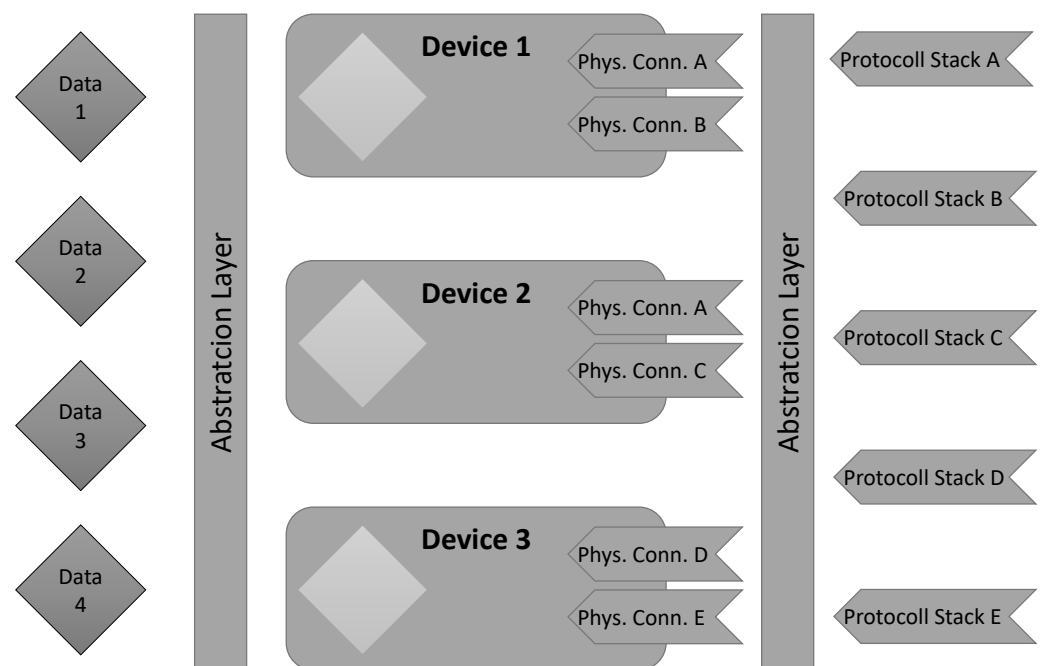


Figure 2. Implementation structure of SBT devices.

The implementation approach leads to one or several devices that can be integrated with an IoT platform. Figure 3 provides an overview of common IoT environments.

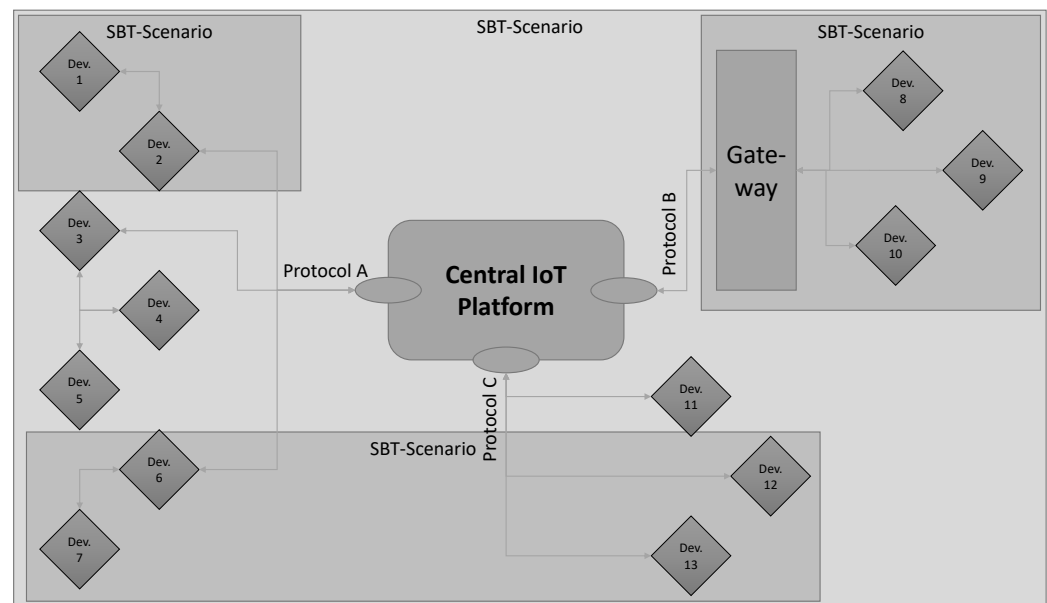


Figure 3. Possible use case scenarios of SBT.

Furthermore, potential applications of SBT are highlighted in this figure. The SBT can either be used to create the highlighted parts and thus be used for retrofitting or extending existing networks. Alternatively, the SBT can be used to create novel IoT environments and is able to provide all of the features. Consequently, the device level, syntactic level, semantic level, cross-platform, and cross-domain interoperability according to Noura et al. [23] are satisfied, leaving only the network level interoperability which can be integrated into the SBT at a later point, but is currently not part of the framework. The remainder of this section describes individual aspects of the SBT components in further detail.

4.2. Data Points

Data is the crucial aspect of any IoT environment. Embedded devices are controlled to perform certain actions, depending on conditions, or sense their surroundings and provide this information. Collecting and transferring data is the basis for any activity. Examples are the turning off of a heater if a sensor discovers an open window or the collection of temperature and humidity. In the context of the SBT, data points contain a piece of data in connection with meta-information, such as unit, type and upper and lower bound. Data points are created by the SBT components and transmitted via communication protocols. This approach provides syntactic level interoperability, as any given data can be defined and integrated into devices.

4.3. Communication Protocols

In order to be universally applicable, the SBT supports a large number of communication protocols. This stack is an initial set-up that will be extended continuously. As a general concept, the communication protocol is abstracted in a way that interfaces take data points, wrap them in packets of the corresponding communication protocol and transmit them via a hardware connector. This approach ensures independence of data points from the communication aspect. The same data point can be created, sent or received on different devices via different protocols. This approach provides device level interoperability. As protocols are available as functional modules, the implementation time is reduced as well. Currently, the following protocols are implemented:

- WiFi
- TCP/IP
- Bluetooth
- I²C
- SPI
- UART

This collection shows a focus on microprocessor-boards and their communication. However, the implementation of protocols from IIoT scenarios, such as Modbus, CAN, and Profibus are currently pending.

4.4. IoT Platform

In IoT scenarios, the dashboard plays a crucial role in data collection, preparation, storage, analysis and visualisation on one hand. On the other hand, devices are commonly controlled via a dashboard platform. This dashboard connects the individual devices, stores and aggregates the information they provide and forms decisions based on the data that is translated to actionable commands. As discussed in Section 2, a multitude of publicly and commercially available solutions focuses on the central platform for control and collection. In the SBT, a central platform is tightly integrated with the devices. The data points are exchanged and integrated into the control logic in the platform. Interfaces for visualisation, data storage, and analysis are provided. In general, this versatile platform allows for semantic level, cross-platform, and cross-domain interoperability, as domains and platforms can be integrated and data can be processed with a semantic understanding.

The implementation solutions for these components are discussed in the next section.

5. Implementation

This section discusses how the components of the SBT architecture, as presented in the previous section, are implemented. The basis of any IoT framework is data, while the backbone is a platform interconnecting devices and making sense of the data. Consequently, communication is the third vital aspect, as devices need to communicate and data needs to be transmitted.

5.1. Data

Data is stored using Google Protocol Buffers [57]. On their website, they are labelled a “language-neutral, platform-neutral extensible mechanism for serializing structured data”. As IoT devices most commonly provide structured data, i.e., a value, a unit, and additional information, such as boundaries, they are easily encoded as Protocol Buffer elements. An advantage of Protocol Buffers is their platform- and software-independence. Currently, the SBT framework is used for embedded devices that are implemented using C and C++. However, the same data can be used in apps on mobile applications or in desktop tools. Furthermore, Protocol Buffer provide a small fingerprint regarding data size which is especially useful in low-latency communications. An exemplary description of a data point can be found in Listing 1.

Listing 1. Exemplary Protocol Buffer definition.

```
message Data {
    required name = 1;
    optional unit = 2;
    required value = 3;
}
```

These definitions are created once for every data point of interest that is to be integrated into and used inside the IoT environment. In the context of the SBT project, data point definitions were created to generalise the data embedded devices are working on. These data points are consequently used and stored in the IoT platform. In the course of the

project, relevant data points were defined to describe common sensor and actuator values, such as temperature, brightness, or current. These data points were shared between all participating devices as well as the IoT platform.

5.2. Platform

A crucial part of IoT environments is a platform that collects, analyses and orchestrates data. Data in this context describes physical environmental conditions, such as temperatures or actuators, such as light switches. This data from distributed entities is gathered and combined to create a meaningful result. For the SBT framework, Node-RED [58] was chosen. It acts as a middleware to connect entities and perform computations. Its application to industrial systems has been, for example, discussed by Nitulescu and Korodi [59] as well as Folgado et al. [60]. The benefits of Node-RED are its easy to use interface, its flexibility, and its potential to be extended with custom functionality. It is lightweight and supports several hardware platforms. It provides a large set of supported functions and communication protocols. Additionally, implementing a semantic representation of the data points by integrating the Protocol Buffer descriptions was a simple task. Node-RED allows the creation of flows that connect inputs from distributed devices to outputs. An exemplary flow is shown in Figure 4.

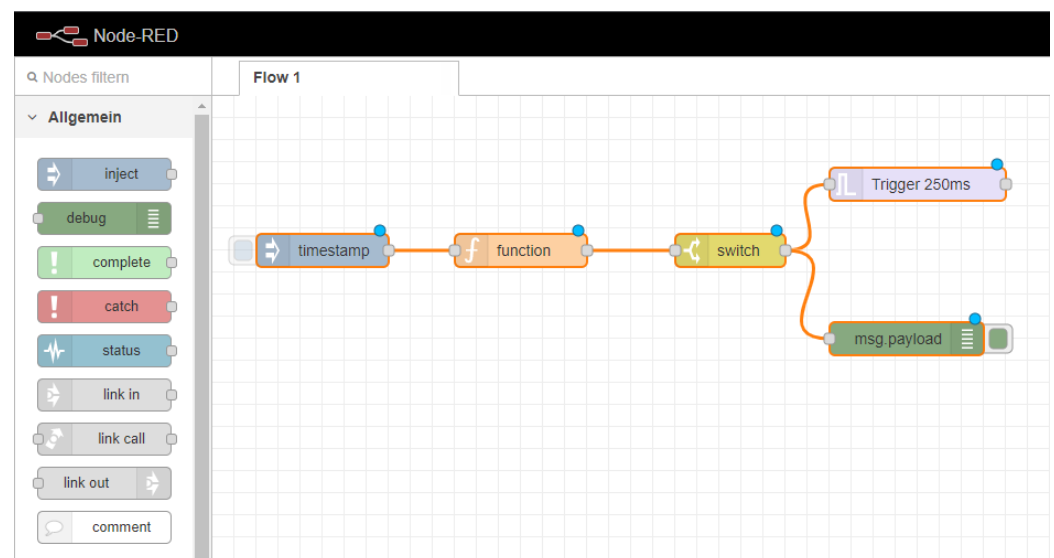


Figure 4. Exemplary flow in Node-RED.

Node-RED furthermore provides means to extract data and store it persistently, for later analysis and visualisation. For the SBT, InfluxDB in combination with Grafana were used, as they both are tailored for time series and the serial values in IoT environment can easily be represented as a time series. Due to the versatile nature of Node-RED, sending time series data into InfluxDB is possible independent of the manner of hosting the database. In our case, the InfluxDB database was located on the same system as Node-RED. However, as a web service, it can be hosted on premise as well as the cloud. Since Node-RED provides connectors for commonly used protocols, it was easy to connect SBT devices to the Node-RED platform. Communication of the SBT framework is discussed in the next subsection. This platform is currently hosted on a Raspberry Pi 3. In the context of the SBT-project, Node-RED modules for collecting and working on the data generated by the embedded devices were implemented. Furthermore, logic to compute output signals that could be translated to actions by the IoT devices was developed. The concept of a Node-RED platform was taken and extended with custom functionality, namely additional function blocks for reading, decoding, encoding, and writing data points to and from the embedded devices. This feature enabled the IoT platform to orchestrate the embedded devices.

5.3. Communication

Communication between machines and devices occurs following specific protocols. These protocols differ depending on the application area. Industrial applications employ certain established protocols, such as Modbus TCP/IP, Profinet, and OPC UA. In the automotive area, CAN is widely used. For home automation and IoT, MQTT, M-Bus, KNX, and Ethernet-based protocols are used for wired communication. In wireless use cases, Zig-Bee, LoRaWAN, and, more recently Thread as the protocol of Matter, are well-established. MQTT can be considered as the de facto standard for IoT applications. These protocols are provided by hardware drivers that correctly employ the hardware interfaces of a device, as well as by software stacks that implement the protocol to ease the integration. For the SBT framework, an interface between data and communication protocol was designed. Thus, the application specific code can hand over data to an abstracted communication interface, while the concrete implementation is device-specific. As a consequence, application-specific code can be re-used by changing the device and corresponding software stack. This layer of abstraction increases the usability and ease of use, while also providing benefits regarding the time to market. Most embedded hardware boards provide libraries with drivers for a given communication stack. The advantage of SBT was to implement and provide a layer of abstraction by creating a wrapper with interfaces that translates between the communication drivers and the data points. This allows for a seamless change of hardware platform, as the implemented wrappers provide the same interface for the application, i.e., the data points, while translating to all common hardware drivers.

5.4. Putting It All Together

Finally, the previously discussed parts of the SBT framework are integrated. Zephyr [22] is used for this integration. Because of its support for a wide variety of embedded devices from different vendors, Zephyr provides support for many different hardware layouts. The interface used to communicate data points to the protocol layer allows for easy integration of the SBT framework into Zephyr. This interfacing, allowing the use of Zephyr as well as use of other embedded programming languages, is a benefit of the SBT project. Every interaction with a hardware driver is abstracted so that a programmer does not have to implement it anew for novel hardware. Furthermore, the abstracted data points can be used in an architecture- and programming language-independent way. In general, the data points are defined for the application using the Google Protocol Buffer, since they are architecture and code agnostic. The communication protocols are implemented from scratch, using common drivers. Furthermore, they are set up with an abstraction layer so that the communication protocols can be set up to work with any given data point on any supported hardware. This is a contribution of SBT. Node-RED is used as an orchestration platform, with custom-made modules for data handling and storage. All components are sufficiently lightweight in terms of resource usage that they are feasible for IoT-environments.

6. Evaluation and Demonstrator Use Cases

In order to evaluate the usefulness and applicability of the SBT framework, two demonstrator scenarios were implemented. A smart home demonstrator was set up using only SBT components. Furthermore, an industrial use case was considered. Here, an existing model factory was extended with additional sensors for Condition Monitoring (CM) and obtaining process information.

6.1. Smart Home Demonstrator

The smart home demonstration environment implements commonly used smart home functionality in a doll house. Smart home functionality as such is not a novel feat. However, the approach in setting up and implementing the features is of interest. The following criteria were evaluated:

- Transferability of functionality
- Time
- Reusability

A comparison with traditional implementation of embedded devices is intended. In this doll house, as shown in Figure 5 the following sensors are physically implemented:

- two Photovoltaic (PV) modules,
- five temperature sensors, and
- one motion detector.

The following actuators are integrated to control the smart home environment:

- five heating elements, simulating heaters,
- two sets of WS2812 NeoPixel LEDs, simulating television and bathtub,
- five LEDs, simulating lightbulbs, and
- one set of LEDs, simulating an oven

Each of those sensors and actuators was described by a data point containing information such as value and unit. This definition was reusable between devices and IoT platform.



Figure 5. Picture of the employed doll house.

These elements were controlled by four embedded devices, two ESP32 micro-controllers that were connected to the IoT platform via Ethernet, one nrf9160 micro-controller, that was connected to an ESP32 via Bluetooth that, in turn, routed communication to and from the platform. Finally, an STM micro-controller Nucleo-L073RZ, that was connected to the other ESP32 via UART, was used. The sensors and actuators were distributed among these four micro-controllers. An overview of this architecture is shown in Figure 6.

From a technical perspective, this architecture is more complex than necessary. However, the aim of this demonstrator scenario was to create complexity. A central IoT platform based on Node-RED was successfully used to control four distributed devices, of which only two had direct access to the platform. A PID controller was implemented for the heating elements, combining temperature values from one physical device with output values from another. Conditions and connections of sensor values and actuator were created in a simple fashion, due to the unified data point-structure. In the course of the demonstrator scenario, several micro-controllers were used. Exchange of devices was simple and quick due to the abstraction layer, provided by the SBT framework, as well as the underlying Zephyr toolchain. The following findings were derived. The base code remained the same. Also the data point definitions were used on every device. The implementation had to be done only once. That means Zephyr provided a base for any feature

implementation. These features were dependent on data points providing information about sensor values and setting actuator targets. The logic was created on the Node-RED platform. Generally, the same code could run on boards of different vendors, with only the pin-layout needing to be adapted. In Zephyr, this can be done with an overlay file that maps logic pin descriptions to the physical pins. Additionally, communication stacks could be integrated into the solutions and were wrapped with standardised interfaces. That allowed transfer of functionality between communication channels. Generally, the SBT functions as a building toolbox that connects hardware drivers, an IoT platform and expressive data point definitions to functionality with few manual labour. The time to change a micro-controller board while maintaining the same functionality was considerably less than a reimplementation would have required.

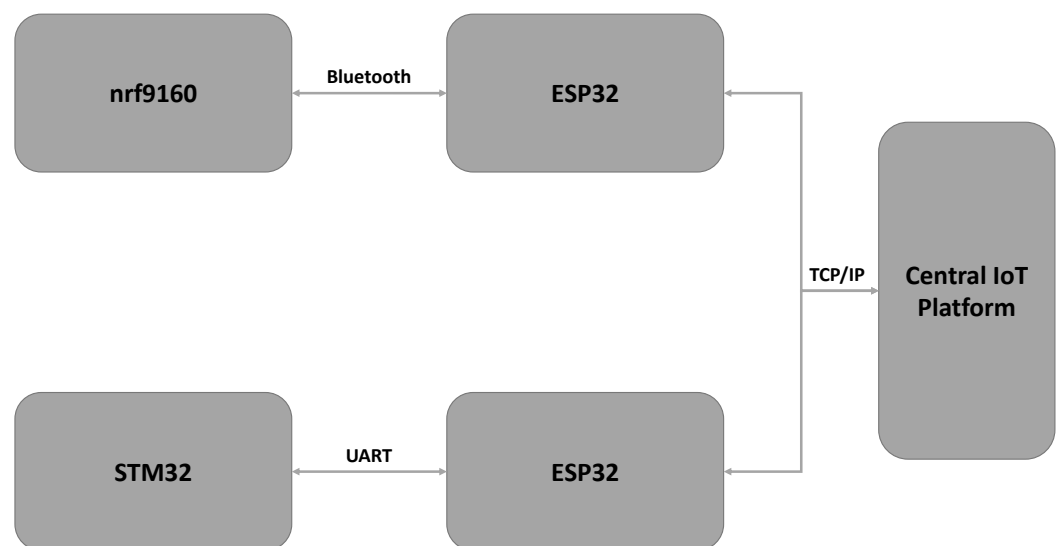


Figure 6. Micro-controller architecture in the smart home demonstrator scenario.

6.2. Smart Factory Demonstrator

In contrast to the smart home demonstrator, the smart factory demonstrator is a so-called brown-field scenario. Such scenarios are often found in practice, as industrial environments commonly contain legacy devices [61,62]. These devices are used for specific steps in processing. They are spatially distributed and difficult to upgrade as upgrades are mostly rolled out via physical medium [61,62]. Legacy and proprietary protocols are used that need to be integrated into the environmental context [61,62]. Due to the cost of downtimes, these machines often run around the clock and are seldomly maintained or updated from a software-perspective. This fact in conjunction with the increase in interconnectivity in industrial areas, creating the IIoT, creates an abundance of vulnerabilities [16,61].

For this demonstrator, a fischertechnik smart factory was employed with the pre-implemented process scenario [63]. It consists of a high-bay warehouse, a crane with vacuum grabber for moving the materials around, a processing as well as a sorting station. Material can be delivered to and from the factory. A picture of this smart factory is shown in Figure 7.

For the smart factory demonstrator, the available system model from fischertechnik was employed to emulate a brown-field scenario. This system uses a Siemens S7-1500 to take in, store, process, and deliver materials. This Programmable Logic Controller (PLC) provides OPC UA-support. OPC UA is an industrial data model that provides connectivity and descriptions of devices and allows, among other features, for M2M communication. An Asset Administration Shell (AAS) can be integrated. This OPC UA functionality was used to notify the SBT dashboard of changes in the output connectors of the PLC. Additionally, an SBT component was set up that measured the electrical current consumed by different parts of the factory. With this sensor-fusion approach, deviations of expected behaviour can

be discovered. The electrical current sensors were controlled by an ESP32 micro-controller, which was integrated into the system. Data points describing the current in terms of value and unit were used to generalise monitoring. The features of the SBT allow for the integration of OPC-UA values and the retrofit of external sensors, such as the ESP32 used in this scenario. This is especially interesting as the base functionality of the smart factory was already implemented on the PLC. The possibility to integrate that into an IIoT with standardised interfaces enables flexibility. To simulate wear of devices, rubber bands were employed to mimic increasing mechanical load as it occurs in practice when lubrication of housing of rotating parts degrades, or when the tools grow dull due to wear and require more force. While the OPC UA state clearly shows that the correct outputs for the processing steps were set, the electrical current sensor is capable of differentiating between zero, one, and two rubber bands. A comparison is shown in Figure 8. It shows that the current required to move a motor is nearly linearly dependant on the number of rubber bands added to the motor. The addition of rubber bands in this case simulates wear of a motor due to degradation of an attached tool or the lubricant used.

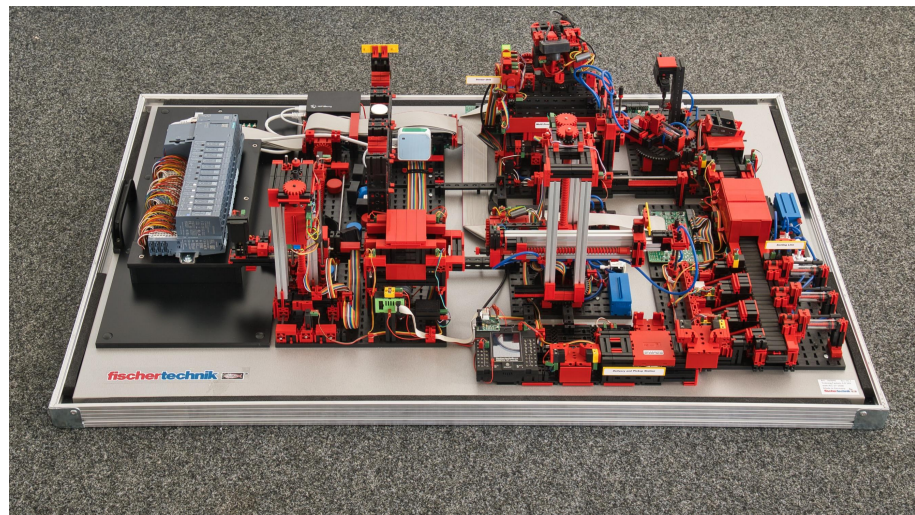


Figure 7. Picture of fischertechnik smart factory.

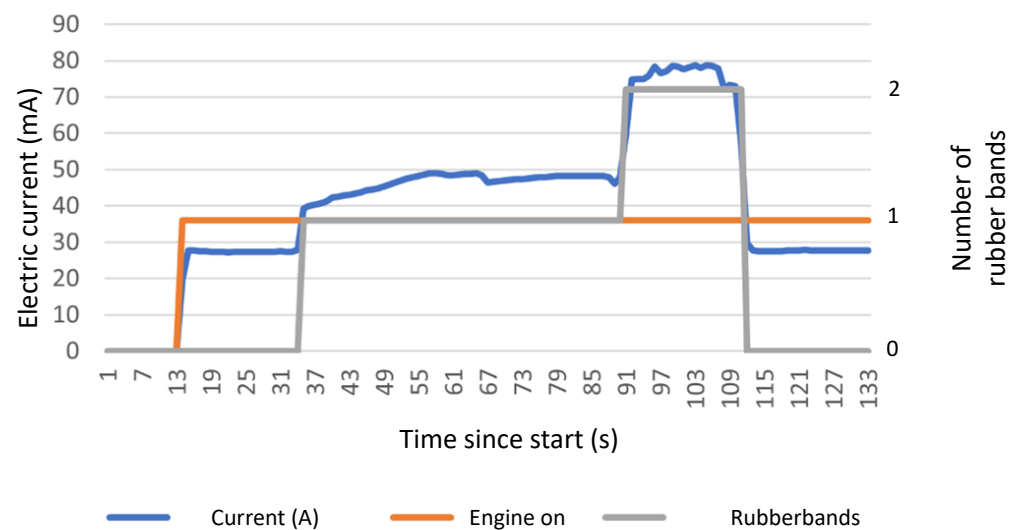


Figure 8. OPC UA state as well as current sensor with zero, one, and two rubber bands influencing the process.

That indicates the value of retrofit sensors and shows their usability. A framework to quickly and easily set up IIoT sensor networks can prove significantly useful in detecting and predicting faults and, consequently, leading to stable operation.

7. Conclusions

In this work, an SBT was introduced. Its contribution is twofold: On the one hand, a framework for quick, easy and versatile setup of partial or complete IoT networks is provided for programmers. On the other hand, a flexible environment is created that can serve as a stand-alone IoT network as well as an extension to existing ones is provided to users. It allows an easy set-up of IoT and IIoT devices as well as platforms to control these devices. The integrated support of commonly used communication protocols with the capabilities for easy extension enable flexible creation and extension of IoT and IIoT environments. As data, its creation, transport, and usage is a core concept of IoT environments, the SBT provides a unified data model based on Google Protocol Buffers. An integration of the data and communication model into the Zephyr-environment creates an easy-to-use framework for quickly and efficiently setting up IoT environments. Standardised interfaces allow seamless integration and extension. Every level of interoperability according to Noura et al. [23], except for network level interoperability, which can easily be integrated as well, is provided. In contrast to other existing solutions, this framework provides value to programmers and users alike.

This concept was evaluated in two application scenarios, a smart home environment as well as an emulated smart factory. According to a non-representative survey of developers participating in this project, the solution was able to reduce the time for set-up and implementation of functionality by factor four on the hardware-side. The intended functionality could be easily set-up and ported between different devices with the presented framework. Implementation of business logic is easily separable from the device control. Control and data storage are integrated into the framework and provide means for visualisation and analysis of data. The implementation of abstraction layers was successfully addressed and enabled seamless transfer of application code between devices. The integration into an IoT platform was achieved by defining and standardising data points as Google Protocol Buffer elements, allowing fast creation and re-use of elements. Computations on the IoT platform were performed using these data points by functions specifically implemented during the project. Wrapper functions were developed to provide standardised interfaces and enable data exchange from physical sources as well as hardware platforms.

Next steps in the development of the SBT include the integration of additional communication protocols, an increase in capabilities for data analysis and an extension of the framework to industrial devices, such as PLCs and others. Due to the different setup approach of PLCs, the adaption will have to take novel features into account, as setting up PLCs for industrial applications is not a trivial task.

Funding: This work has been supported by the Investment and Structure Bank (ISB) Rhineland-Palatinate (Foerderkennzeichen P1SZ26, InnoTop). The authors alone are responsible for the content of this paper.

Data Availability Statement: The datasets presented in this article are not readily available because they contain sensitive information. Requests to access the datasets should be directed to the first author.

Conflicts of Interest: Author Simon D. Duque Anton was employed by the company comlet Verteilte Systeme GmbH. The author declares no conflicts of interest.

References

1. Leukert, B.; Kubach, T.; Eckert, C.; Tsutsumi, K.; Crawford, M.; Vayssiere, N.; Kandathil, E.T.; Kubach, U.; Majumdar, A.; Southall, A.; et al. *White Paper: IoT 2020: Smart and Secure IoT Platform*; Technical Report; International Electrotechnical Commission (IEC): Geneva, Switzerland, 2020.
2. Chui, M.; Löffler, M.; Roberts, R. McKinsey Quarterly: The Internet of Things. 2010. Available online: <https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-internet-of-things> (accessed on 23 June 2024).
3. Fortune Business Insights™. *Internet of Things (IoT) Market Size*; Technical Report FBI100307; Fortune Business Insights™: Pune, India, 2022.
4. Alaa, M.; Zaidan, A.; Zaidan, B.; Talal, M.; Kiah, M. A review of smart home applications based on Internet of Things. *J. Netw. Comput. Appl.* **2017**, *97*, 48–65. [\[CrossRef\]](#)
5. Patru, I.I.; Carabas, M.; Barbulescu, M.; Gheorghe, L. Smart home IoT system. In Proceedings of the 2016 15th RoEduNet Conference: Networking in Education and Research, Bucharest, Romania, 7–9 September 2016. [\[CrossRef\]](#)
6. Kim, T.H.; Ramos, C.; Mohammed, S. Smart City and IoT. *Future Gener. Comput. Syst.* **2017**, *76*, 159–162. [\[CrossRef\]](#)
7. Gaur, A.; Scotney, B.; Parr, G.; McClean, S. Smart City Architecture and its Applications Based on IoT. *Procedia Comput. Sci.* **2015**, *52*, 1089–1094. [\[CrossRef\]](#)
8. Zamora-Izquierdo, M.A.; Santa, J.; Martínez, J.A.; Martínez, V.; Skarmeta, A.F. Smart farming IoT platform based on edge and cloud computing. *Biosyst. Eng.* **2019**, *177*, 4–17. [\[CrossRef\]](#)
9. Dagar, R.; Som, S.; Khatri, S.K. Smart Farming—IOT in Agriculture. In Proceedings of the 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 11–12 July 2018. [\[CrossRef\]](#)
10. Butun, I. (Ed.) *Industrial IoT*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020. [\[CrossRef\]](#)
11. Siozios, K.; Anagnostos, D.; Soudris, D.; Kosmatopoulos, E. (Eds.) *IoT for Smart Grids*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019. [\[CrossRef\]](#)
12. Caro, F.; Sadr, R. The Internet of Things (IoT) in retail: Bridging supply and demand. *Bus. Horiz.* **2019**, *62*, 47–54. [\[CrossRef\]](#)
13. Dlamini, N.N.; Johnston, K. The use, benefits and challenges of using the Internet of Things (IoT) in retail businesses: A literature review. In Proceedings of the 2016 International Conference on Advances in Computing and Communication Engineering (ICACCE), Durban, South Africa, 28–29 November 2016. [\[CrossRef\]](#)
14. Selvaraj, S.; Sundaravaradhan, S. Challenges and opportunities in IoT healthcare systems: A systematic review. *SN Appl. Sci.* **2019**, *2*, 139. [\[CrossRef\]](#)
15. Duque Anton, S.D. *Anomaly Detection in Industry*; Verlag Dr. Hut: Munich, Germany, 2021.
16. Duque Anton, S.D.; Fraunholz, D.; Krohmer, D.; Reti, D.; Schneider, D.; Schotten, H.D. The Global State of Security in Industrial Control Systems: An Empirical Analysis of Vulnerabilities Around the World. *IEEE Internet Things J.* **2021**, *8*, 17525–17540. [\[CrossRef\]](#)
17. Ramaswami, V.; Inoue, T.; Fraval, G.; Salmanpour, M.; Walters, D. Reduce Risk, Time to Market, and Cost During IoT Hardware Development. *AWS Partner Network (APN) Blog*, 24 September 2021.
18. Kovacs, E. Reuse of Cryptographic Keys Exposes Millions of IoT Devices: Study. *Security Week*, 25 November 2015.
19. Wegner, P. IoT Platform Companies Landscape 2021/2022: Market Consolidation Has Started. 2021. Available online: <https://iot-analytics.com/iot-platform-companies-landscape/> (accessed on 23 June 2024).
20. Arduino. What Is Arduino? 2022. Available online: <https://www.arduino.cc/> (accessed on 23 June 2024).
21. MIKROE. Time-Saving Embedded Tools. 2022. Available online: <https://www.mikroe.com/> (accessed on 23 June 2024).
22. Zephyr. The Zephyr Project Strives to Deliver the Best-in-Class RTOS for Connected Resource-Constrained Devices, Built to Be Secure and Safe. 2022. Available online: <https://zephyrproject.org/> (accessed on 23 June 2024).
23. Noura, M.; Atiquzzaman, M.; Gaedke, M. Interoperability in Internet of Things: Taxonomies and Open Challenges. *Mob. Netw. Appl.* **2018**, *24*, 796–809. [\[CrossRef\]](#)
24. Braten, A.E.; Kraemer, F.A.; Palma, D. Autonomous IoT Device Management Systems: Structured Review and Generalized Cognitive Model. *IEEE Internet Things J.* **2021**, *8*, 4275–4290. [\[CrossRef\]](#)
25. Cheruvu, S.; Kumar, A.; Smith, N.; Wheeler, D.M. IoT Frameworks and Complexity. In *Demystifying Internet of Things Security*; Apress: New York, NY, USA, 2019; pp. 23–148. [\[CrossRef\]](#)
26. Aïssaoui, F.; Berlemont, S.; Douet, M.; Mezghani, E. A Semantic Model Toward Smart IoT Device Management. In *Web, Artificial Intelligence and Network Applications; Advances in Intelligent Systems and Computing*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 640–650. [\[CrossRef\]](#)
27. Privat, G. Extending the Internet of Things. *Commun. Strateg.* **2012**, *3*, 101–119.
28. Fortino, G.; Rovella, A.; Russo, W.; Savaglio, C. Towards Cyberphysical Digital Libraries: Integrating IoT Smart Objects into Digital Libraries. In *Internet of Things*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 135–156. [\[CrossRef\]](#)
29. Uviase, O.; Kotonya, G. IoT Architectural Framework: Connection and Integration Framework for IoT Systems. *Electron. Proc. Theor. Comput. Sci.* **2018**, *264*, 1–17. [\[CrossRef\]](#)
30. Chen, J. Devify: Decentralized internet of things software framework for a peer-to-peer and interoperable IoT device. *ACM SIGBED Rev.* **2018**, *15*, 31–36. [\[CrossRef\]](#)

31. Sun, L.; Li, Y.; Memon, R.A. An open IoT framework based on microservices architecture. *China Commun.* **2017**, *14*, 154–162. [CrossRef]
32. Mavromatis, A.; Colman-Meixner, C.; Silva, A.P.; Vasilakos, X.; Nejabati, R.; Simeonidou, D. A Software-Defined IoT Device Management Framework for Edge and Cloud Computing. *IEEE Internet Things J.* **2020**, *7*, 1718–1735. [CrossRef]
33. Maloney, M.; Reilly, E.; Siegel, M.; Falco, G. Cyber Physical IoT Device Management Using a Lightweight Agent. In Proceedings of the 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, GA, USA, 14–17 July 2019. [CrossRef]
34. Großmann, M.; Illig, S.; Matějka, C.L. SensIoT: An Extensible and General Internet of Things Monitoring Framework. *Wirel. Commun. Mob. Comput.* **2019**, *2019*, 4260359. [CrossRef]
35. Calderoni, L.; Magnani, A.; Maio, D. IoT Manager: An open-source IoT framework for smart cities. *J. Syst. Archit.* **2019**, *98*, 413–423. [CrossRef]
36. Balachandar, S.; Chinnaiyan, R. Centralized Reliability and Security Management of Data in Internet of Things (IoT) with Rule Builder. In *International Conference on Computer Networks and Communication Technologies*; Springer: Singapore, 2018; pp. 193–201. [CrossRef]
37. Gelogo, Y.E.; Hwang, H.J.; Kim, H.K. Internet of Things (IoT) Framework for u-healthcare System. *Int. J. Smart Home* **2015**, *9*, 323–330. [CrossRef]
38. Burger, A.; Cichowskyj, C.; Schmeißer, S.; Schiele, G. The Elastic Internet of Things - A platform for self-integrating and self-adaptive IoT-systems with support for embedded adaptive hardware. *Future Gener. Comput. Syst.* **2020**, *113*, 607–619. [CrossRef]
39. Perumal, T.; Datta, S.K.; Bonnet, C. IoT device management framework for smart home scenarios. In Proceedings of the 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE), Osaka, Japan, 27–30 October 2015. [CrossRef]
40. Microsoft. Azure IoT. 2022. Available online: <https://azure.microsoft.com/en-us/solutions/iot/#overview> (accessed on 23 June 2024).
41. IBM. IBM Cloud. 2022. Available online: <https://www.ibm.com/cloud> (accessed on 23 June 2024).
42. GE Digital Solutions. Predix Platform. 2022. Available online: <https://www.ge.com/digital/documentation/predix-platforms/index.html> (accessed on 23 June 2024).
43. Amazon. AWS IoT. 2022. Available online: https://aws.amazon.com/iot/?nc1=h_ls (accessed on 23 June 2024).
44. Bilgeri, D.; Brandt, V.; Lang, M.; Tesch, J.; Weinberger, M. *White Paper: The IoT Business Model Builder*; Resreport; Bosch IoT Lab: St.Gallen, Switzerland, 2015.
45. Twilio. DEVICE BUILDER PLATFORMS—IOT Infrastructure for Lifelong Device Connections. 2022. Available online: <https://www.twilio.com/iot/device-builder-platform> (accessed on 23 June 2024).
46. Openremote. 100% Open Source IoT Platform. 2022. Available online: <https://www.openremote.io/product/> (accessed on 23 June 2024).
47. MiMove—Middleware on the Move. Towards a Framework for Developing Extensible IoT Applications. 2018. Available online: <https://mimove.inria.fr/zefxis/> (accessed on 23 June 2024).
48. Macchina. Build, Deploy and Remotely Manage IoT Edge Devices and Applications. 2022. Available online: <https://macchina.io/> (accessed on 23 June 2024).
49. DeviceHive. IoT Made Easy. 2022. Available online: <https://devicehive.com/#home> (accessed on 23 June 2024).
50. DSA. The Open Source Platform & “Toolkit” for Internet of Things Devices, Services and Applications. 2022. Available online: <https://iot-dsa.org/> (accessed on 23 June 2024).
51. Kaa. Start Building Your Solution with Kaa Enterprise Iot Platform. 2022. Available online: <https://www.kaaiot.com/#configuration-management> (accessed on 23 June 2024).
52. openHAB. Empowering the Smart Home. 2022. Available online: <https://www.openhab.org/> (accessed on 23 June 2024).
53. zetta. An API-First Internet of Things Platform. 2022. Available online: <https://www.zettajs.org/> (accessed on 10 October 2023).
54. Kulkarni, P. Top 8 Challenges in IoT Development—And How to Overcome Them. 2022. Available online: <https://bytebeam.io/blog/top-8-challenges-in-iot-developmentand-how-to-overcome-them-cl5m9r7e1381521olz82atno32> (accessed on 23 June 2024).
55. Emorphis Technologies. IoT Application Development Challenges. 2019. Available online: <https://blogs.emorphis.com/iot-application-development-challenges/> (accessed on 23 June 2024).
56. Hackbarth, K. The Three Challenges of IoT Solution Development. 2016. Available online: <https://blog.bosch-si.com/internetofthings/the-three-challenges-of-iot-solution-development/> (accessed on 10 October 2023).
57. Google. Protocol Buffers. 2022. Available online: <https://developers.google.com/protocol-buffers> (accessed on 23 June 2024).
58. Node-RED: Low-Code Programming for Event-Driven Applications. 2022. Available online: <https://nodered.org/> (accessed on 23 June 2024).
59. Nitulesu, I.V.; Korodi, A. Supervisory Control and Data Acquisition Approach in Node-RED: Application and Discussions. *IoT* **2020**, *1*, 5. [CrossRef]
60. Folga, F.J.; Calderón, D.; González, I.; Calderón, A.J. Review of Industry 4.0 from the Perspective of Automation and Supervision Systems: Definitions, Architectures and Recent Trends. *Electronics* **2024**, *13*, 782. [CrossRef]

61. Serror, M.; Hack, S.; Henze, M.; Schuba, M.; Wehrle, K. Challenges and Opportunities in Securing the Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2021**, *17*, 2985–2996. [[CrossRef](#)]
62. Chowdhury, A.; Raut, S.A. Benefits, challenges, and opportunities in adoption of industrial IoT. *Int. J. Comput. Intell. IoT* **2019**, *2*, 822–828.
63. fischertechnik. Training Factory Industry 4.0 24V Complete Set with PLC S7-1500. 2023. Available online: <https://www.fischertechnik.de/en/products/industry-and-universities/training-models/560841-training-factory-industry-4-0-24v-with-plc-connection-board> (accessed on 23 June 2024).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.